Red Hat

published by Red Hat Developer

THE MODERN DEVELOPER

INDEX

Organizations of all sizes and shapes globally are increasingly digitally transforming to solve business problems, stay competitive, and improve customer experience and profitability. As part of this transformation, the job of developers has evolved as well. Developers now play a critical role in the digital transformation journey by helping businesses deliver faster and meaningful business outcomes.

This shift in focus requires enterprises to revisit the IT landscape to provide an improved and efficient ecosystem for developers to be successful in this new journey. The shift involves not only providing better and more modern tools and technologies, but also changing the operating model and team structure to drive business outcomes as the shared goal. Enterprises have successfully achieved this shift with a combination of Dev(Sec)Ops practices, automation, cloud-native development, and Platform as a Service (PaaS) deployments. In this microbook, we will see how enterprise IT has evolved from business support functions to becoming a key enabler of next generation digital native businesses, and how the evolution has touched every aspect of IT personnel, systems, and processes.

IDC predicts that the percentage of large organizations that deploy code to production daily will increase from 5% in 2021 to 70% in 2025 as a result of widespread implementation of mature DevOps practices. Furthermore, as traditional DevOps automation and processes are disrupted by Kubernetes and cloud-native development, the firm estimates that by 2024, 35% of DevOps adopters will embrace more streamlined GitOps automation processes.[1]

How did we get here?

We can classify the gradual evolution of applications development and IT operations into three distinct phases, from the early mainframe days through the early Web-driven "dotcom era" to modern cloud-native development practices. The following subsections summarize each phase.

The early days

For most of its history, digital technology was used by enterprises to digitize the existing business processes with little or no change to the way the businesses ran on a day to day basis. For instance, even after the digitization of banks and public sector industries, the customers still had to visit the businesses in-person and interact with an employee to get things done.

The customer experience changed a little after interactive voice response (IVR) systems gained popularity in the 1980s and 1990s. These telephone-based systems were able to address some basic questions for the customers, thus avoiding time consuming in-person visits. The trend gradually extended to phone support to help customers with some advanced capabilities. But still, at that point, businesses were never under pressure to exploit computer technology to improve the way they delivered services or to find new revenue streams. Businesses could not

provide highly differentiated services with the limited technologies that were available at that time and the cost of experimentation is high. So customer loyalty still bolstered business as usual, and "customer for life" remained the norm.

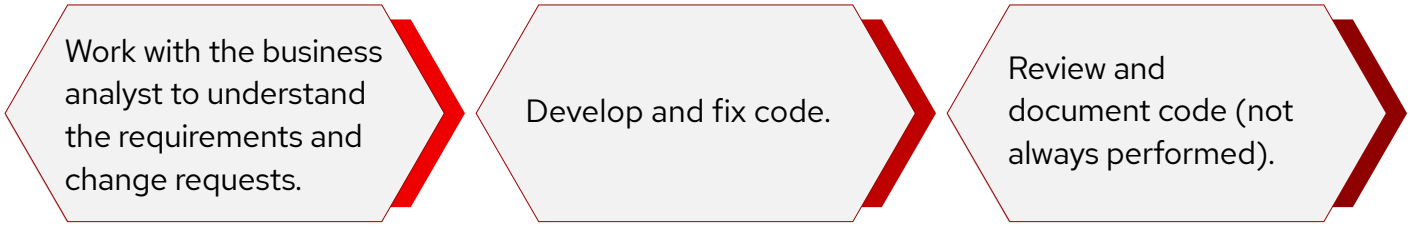
In that environment, IT departments were run as a business support function. The teams built and managed long running applications, typically with waterfall development methodology and long shelf life. Given that these systems supported core business functions, every change or new feature was meticulously tested and validated to make sure no new risks and bugs were introduced that could damage the business. Quality was ensured by creating multiple siloed organizations who had very specific tasks to deliver, each in its own clearly defined swim lane with little to no overlap with other teams. A typical IT team structure at this time involved:

- **Business analysts:** Responsible for capturing and documenting business requirements for the applications.
- **Development team:** Responsible for building and updating applications.
- **QA team:** Responsible for testing applications before it was deployed to production.
- **Build and release management team:** Responsible for building and deploying applications from source code to test, UAT, and production environments.
- **Operations team:** Responsible for building and maintaining infrastructure for the applications.
- **Enterprise and solution architecture team:** Responsible for creating and maintaining architectures that reflected key business, security, and compliance requirements. All applications were required to implement the architecture guidelines to be approved for production deployment.
- **Security and compliance teams:** Responsible for ensuring that the applications and the processes followed organizational security and compliance standards.
- **Application support team:** Responsible for supporting the applications in production.

Each role was distinct and execution was predominantly sequential. The model resulted in long, massive releases. It was not uncommon for these applications to miss the mark for customer or

business expectations, since the business stakeholders could see the application only at the end of the cycle, typically during user acceptance testing.

In this model, the role of developer was limited, as was interaction and participation across teams. The core responsibilities for the developers were:



The tools used were also typically simple: an integrated development environment (IDE) for development, a source code repository, a bug tracking system, and a build management tool. The development team did not generally control or even have visibility into when their code would be released to production, apart from creating run books or play books to help operations teams deploy and manage the applications. Thus, developers typically lagged in understanding the goals and uses of the application or module they contributed to.

Dotcom boom

In the mid to late 1990s, the availability of the internet and the associated "dotcom boom" was the first major step toward changing the way the businesses were run and introducing new ways of doing business with ecommerce. Enterprises were now able to develop new web channels to reach out to their customers and to acquire new customers and revenue streams. This momentum brought in some changes to the application delivery process. Developers took on additional responsibilities with test-driven development (TDD) and Agile development practices, which helped organizations reduce the time to production significantly (from years to months).

However, the team structure and the responsibilities remained siloed, and the applications were still monolithic, limiting the benefits that the Agile and TDD approaches could bring to the table. Public clouds didn't yet exist and open source was not entirely pervasive in the enterprise yet, so enterprises were not generally pressured externally to become more innovative and agile. (The exceptions were financial services and other industries who were given more freedom at that time to explore new offerings for their customers, thanks to deregulation, encouraging them to become more innovative and agile.) Owning state-of-the-art data centers was still a key differentiating factor toward which most enterprises aimed.

Modern era

With the introduction of public cloud in mid 2006, the once high bar of entry to large-scale computer operation was significantly lowered, enabling a lot of innovative startups with shoestring budgets to explore new opportunities and eventually challenge established enterprises. This phase also brought in new development approaches: DevOps and later DevSecOps, Site Reliability Engineering (SRE), cloud-native microservices, CI/CD and Platform Engineering. This exploited the readily available, on-demand cloud infrastructure and maximized collaboration between the various teams to deliver faster business outcomes.

These startups are not structured in silos, and focus on taking their ideas to the market at a faster phase by iterating upgrades based on customer feedback loops. This new pace required the IT organization to work closely, almost on a daily basis, with the business teams and customers.

The success of this model has created a lasting impact on how the IT teams are built and operated globally, from startups to large enterprises. Under the common umbrella of digital transformation, global enterprises are now embracing DevSecOps, SRE, and cloud-native microservice development as a standard. These practices can maximize developer productivity to address competitive threats as well as explore new opportunities. In order to achieve faster innovation, enterprises are building internal developer platforms for their teams.

"Developers and DevOps professionals continue to be in the hot seat as their organizations or their customers demand that convenient and user-friendly solutions be created and deployed at an even faster rate. We see the industry on the edge of gaining some fascinating new tools that will help accelerate both development and deployment in the years to come, reducing at least some of the pressure to generate more applications more quickly."

- Al Gillen, group VP, Software Development and Open Source, IDC[1]

What is an Internal Developer Platform?

As explained in the previous section, with technology becoming the primary channel to deliver products and services, developer productivity and overall developer experience take the center stage. To bring enhanced capabilities to customers at a faster pace, with higher quality, developers need to focus on far-reaching innovation and leave the logistics to the platform and tools. The goal of faster innovation can be achieved by building an internal developer platform (IDP) that acts as a single point of contact for all developer needs. An IDP is a self-service layer that allows developers to interact independently with their organization's software delivery tools, processes, and setup. The platform enables developers to self-provision environments by abstracting away infrastructure decisions and seamlessly integrates with organization's continuous integration and delivery (CI/CD), application deployment, and application management processes.

However, the developer platform is not just a productivity tool for developers, but also a mechanism for operations teams to align with developers on standards and workflows that support execution, control, and security. Requirements and the associated processes differ from one enterprise to another.

In short, the goal of the IDP is to abstract away all complexities of provisioning infrastructure, building applications, and managing applications in production, letting the developers focus on the “inner loop” of developing applications to meet new business needs quickly and iteratively.

Some key characteristics of the developer platform are:

01 **Self-service provisioning**

02 **State-of-the-art IDE, application platform, frameworks, and tools**

03 **An automated, secure, and streamlined path to production (the software supply chain)**

04 **Security by default**

05 **Automated, declarative security**

06 **Observability for applications (telemetry and monitoring)**

Self-service provisioning

The key driver of developer velocity starts with self-service. The development teams need to be able to provision and scale their environments and development tools whenever they need, without depending on other teams.

Many enterprises now provide developers with internal portals that fulfill this need. The portals are prepopulated with a curated list of infrastructure elements, platforms, and tools ready for developers to provision and consume. The portal is periodically updated with new offerings based on developer feedback and industry trends, and in certain cases can be directly integrated into the IDE via plugins for automated updates. The main aspect of this portal is to provide on-demand access to infrastructure and tools to developers to start development as soon as possible.

State of the art IDE, application platform, frameworks, and tools

As more and more organizations move toward cloud-native development for their digital transformation, identifying an application platform that fits the enterprise's current and future needs is imperative. We believe that containers will be the deployment unit of choice and that Kubernetes platforms are going to be the most suitable choice for most enterprises, given their broad industry adoption and the benefits they provide, such as reducing the complexity of moving working code from the developer IDE to other environments.

Containers and Kubernetes ensure the application's availability and scalability, and use repositories or images to standardize and secure applications as well as tools and libraries. There are many options available, from on-premises, home-grown platforms to public cloud Kubernetes services. Most organizations are already planning an application development and delivery platform that spans multiple cloud providers to maximize the benefits, open access to more options, and eliminate risks.

Development frameworks and tools are also increasingly cloud-based now. These choices differ from organization to organization based on their developer skill sets, existing technology stack, and business needs. However, the best practice is to build a repository that includes a developer IDE, microservice and serverless frameworks, backend tools such as databases, streamlining platform, API management, security and compliance tools, etc. that can be deployed on the enterprise application platform discussed in the previous section. The goal here is to reduce the cognitive load on developers and get them up to speed, so that they can focus on innovation.

It’s important to consider the software supply chain as well. The software supply chain can be defined as the process that takes source code and delivers a working product in an automated and secured pipeline. The pipeline typically packages the source code and its dependencies as a

container, scans the container and codebase for vulnerabilities, stores the container in a secure image repository, performs automated testing, and once all these preliminary steps are successful, deploys the application to the desired environment.

The concept of a software supply chain has been gaining momentum recently. By distinguishing between the inner loop of rapid development and testing and the outer loop of deployment, the concept helps organizations implement an automated and streamlined approach to delivering software.

Figure 1 illustrates the inner loop (on the left) and outer loop (on the right) of a software supply chain. Everything that is required to take the code from the source code repository to production is part of the outer loop. The inner loop can undergo many iterations before the developers invoke the outer loop to deploy the application.

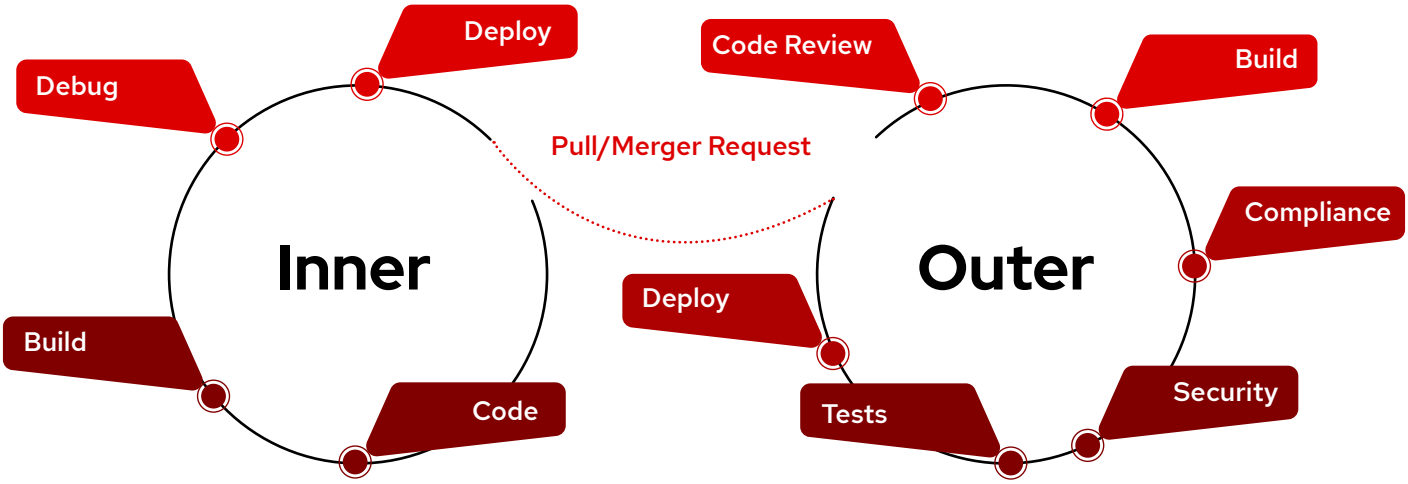


Figure 1: Inner and outer loops of a software supply chain.

As the saying goes, "The happy place for developers is Production." The software supply chain removes all the toil of taking approved code from developers to production as quickly and securely as possible.

The IDP spans both the inner and outer loops, but concentrates on the inner loop. Anything and everything that is needed for developers to be up and running as early as possible and to develop, test, document, and produce high quality software is part of the inner loop. Developers have strong opinions on the tools they want to use which plays a significant role in how the inner loop is constructed. Some of the common elements of this layer include:

01 Developer portal

02 IDE

03 Application development and delivery platform

04 Application frameworks and runtimes

05 Catalog of tools and libraries

The following subsections describe the design considerations and tools that are part of the inner loop, because they apply most closely to developers.

Developer portal

The developer portal is a relatively new concept that is gaining popularity in big enterprises. Developer portals are seen as a way to improve self-service, standardization, and faster onboarding for developers, boosting their innovations. The provision of a portal enables developers to identify and nominate the best technologies for their enterprises in a much more streamlined way than the traditional approach relying on an operations team. A well-constructed developer portal provides developers with at least the following capabilities:

01 Self-service provisioning of infrastructure & development tools

02 Catalog of applications, libraries, third-party tools, etc.

03 (Sometimes) Observability and monitoring tools and log access

The developer portal helps not only developers, but other core teams such as Enterprise Architecture, Operations, Security, and Compliance. Thus, the portal provides a controlled and automated environment that improves the overall performance and efficiency of the overall development lifecycle.

The developer portal should become a single point of access and source of truth for developers to support all their needs. It also can be used as an enforcement layer to address organizational security and compliance needs via a more structured onboarding and offboarding of systems and tools. Optionally, organizations can extend the developer portal to include observability and monitoring tools and log access and auditing, so that the developers have a single point of

access for all systems they are concerned with. It also helps organizations streamline their access management to avoid unauthorized accesses that result in additional security and compliance risks.

Being productive motivates developers. Without the friction, they have time to think creatively and apply themselves - Tim Cochran[2]

One of the best known use cases for a developer portal to improve developer efficiency comes from Spotify. Spotify's developer experience team mapped out the issues that negatively affect the developer experience and reduce productivity. They call those barriers a "negative flywheel," illustrated in Figure 2. The key issues identified are:

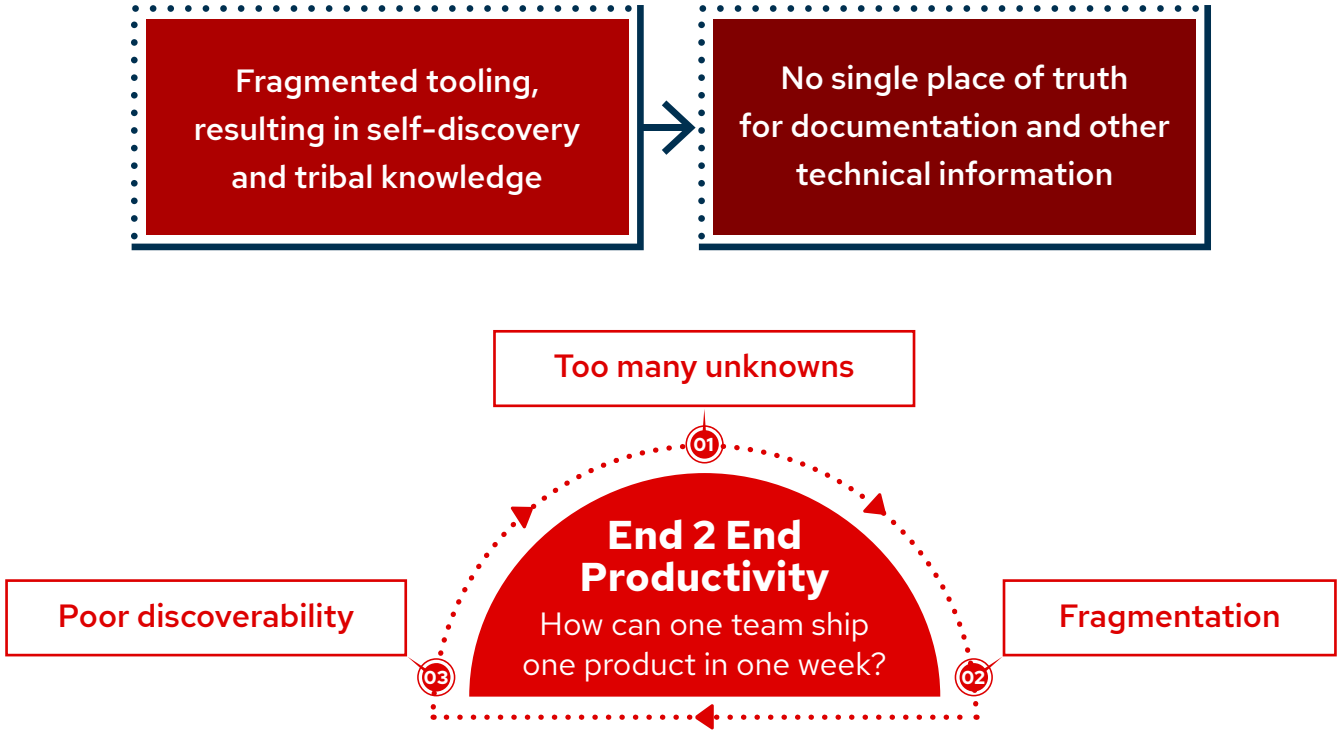


Figure 2: Negative issues creating a negative flywheel.[2]

To mitigate these complexities, Spotify developed a developer portal called *Backstage*. Based on open source software, Backstage exposes all infrastructure products and documentation in one place. This developer portal offers a coherent developer experience and a starting point for engineers to find the information they need without relying on self-discovery, tribal knowledge, and other teams, resulting in faster on-boarding, reduced cognitive load, and improved productivity.

Integrated Development Environment (IDE)

Developers spend most of their time in their IDE. Considering this heavy reliance on the IDE, organizations often suggest heavyweight IDEs that fit well with their internal technology and application landscapes, for developers to install and configure on their desktops and laptops.

One of the objectives of this early generation of IDEs was to provide a self-contained developer eco-system so that the developers would be able to work independently whether or not they were connected to the internet. But this approach introduced lot of complexities in the overall development process, such as:

01 **Incompatible environments:** Since developers were provided with their own sandboxes, they likely customized their local IDEs by adding different frameworks and libraries that they needed, which could create configuration drift and make it difficult to identify and fix problems.

02 **New or duplicate libraries:** Another issue commonly arose when developers introduced duplicate libraries without being aware that similar ones existed in the environment. A developer might also introduce new libraries without getting consensus from other developers, which could introduce additional risks and complexities down the line. This practice of letting developers extend and modify their local environments based on their needs led to bloated and less secure applications. Although most enterprises now have post-development code walkthroughs and testing that captures and addresses most of these issues, it would be better if the incompatibilities and waste can be prevented in the first place.

03 **Mismatched experiences:** Another well-known issue is the "Works for me!" claim heard when code developed by one developer fails to deploy and run on a common shared environment or on another developer's IDE due to incompatibility. The incompatibilities reduce the overall productivity and collaboration among developers. Famously, Docker developed containers to address these very issues. Containers have definitely worked wonders by simplifying the way code is distributed among developers and from one environment to another. However, containers do not address mismatched library versions or missing libraries that do not make it to the container from the developer's device.

04

Cloud deployment: As containers become the primary choice for packaging and distributing applications, tools and languages external to the IDE take on crucial roles in the developers' toolkit. The developers either need to build the containers by providing manifests or employ extra tools to build the images for them.

The problems just cited make it worthwhile to consider a new approach to the IDE in the development environment. One option is a lightweight, modern, browser-based IDE that does not require installation. Organizations who prefer to use their established IDE should explore an application platform that provides extensions and plugins to connect with other platforms. The shared environment will be used by the entire team, eliminating duplicated and mismatched libraries and features. This approach also typically provides a standardized and automated way to build containers, reducing the effort required by multiple developers to build containers locally and deploy them to the development environment.

Application development and delivery platform

A well-engineered application platform is critical to success with the organization's digital transformation, enabling the developers to code and deliver applications faster and more safely to meet the desired business outcomes. With containers becoming the standard unit of deployment, more organizations are building their application platforms on Kubernetes. Other pressures arise from innovations such as 5G, AI/machine learning, and data science, along with data sovereignty and compliance requirements from internal and external stakeholders. Consequently, many enterprises are leaning toward a hybrid cloud application platform instead of building a single data center platform or going with a single cloud provider solution, to avoid vendor lock and to focus better on delivering improved business outcomes. Application platforms should also provide developers with a rich set of preconfigured tools and backend systems to enable faster and safer innovation via a catalog or marketplace and out-of-the-box security and automation capabilities.

The application platform must also provide the features that make the application secure and accessible via load balancing, routing, access control, etc.

Application frameworks and runtimes

Another key aspect of developer platforms is a rich set of application frameworks that enables faster innovation. Polyglot development practices, allowing multiple languages and frameworks, are becoming more and more popular, and are augmented by the modern full-stack approach that combines all levels of the application. The developer platform should include multiple application frameworks and language runtimes and keep growing to accommodate the latest asks from the developers.

The richer the application frameworks you support, the more complex the application build processes are. Technologies such as Buildpacks and Source-to-Image (S2I) are available to standardize and streamline the build process without adding additional development or operations complexities.

Catalog of tools and libraries

Applications do not live in isolation: They need to talk to each other, exchange data, share insights, and pass around security credentials to deliver business capabilities. A developer platform will not be functional without a curated catalog of the tools and libraries required by the development teams to deliver business applications: databases, streaming solutions, messaging queues, third party libraries, Integration tools, etc.

Like application frameworks, the tools catalog will keep evolving, adding the latest technologies and removing tools that are obsolete or no longer needed. Typically, organizations have a highly structured on-boarding and off-boarding process to ensure that the catalog is kept in alignment with organizational security and compliance requirements and to ensure that offerings are not removed prematurely; some might be required for future development. The catalog also ensures that developers now have a single source of truth to provision their backend systems, ensuring faster, streamlined development via self-service without having to depend on the operations team to set up resources. The provisioning of the tools is automated end to end to ensure consistency and speed.

Organizations can also improve this capability by adding reporting and analytics for auditing and compliance purposes.

How can Red Hat help?

One of the key components of digital transformation and DevSecOps is a highly scalable, flexible, and secured application platform that supports containerized, cloud-native applications (Figure 3) across systems of innovation, systems of differentiation, and systems of record. Red Hat, with its open hybrid cloud architecture built with top open source projects such as Red Hat Enterprise Linux and Red Hat OpenShift, delivers these requirements across a hybrid cloud ecosystem from edge to public cloud providers, making it an ideal platform for enterprises across various industries and geographies.

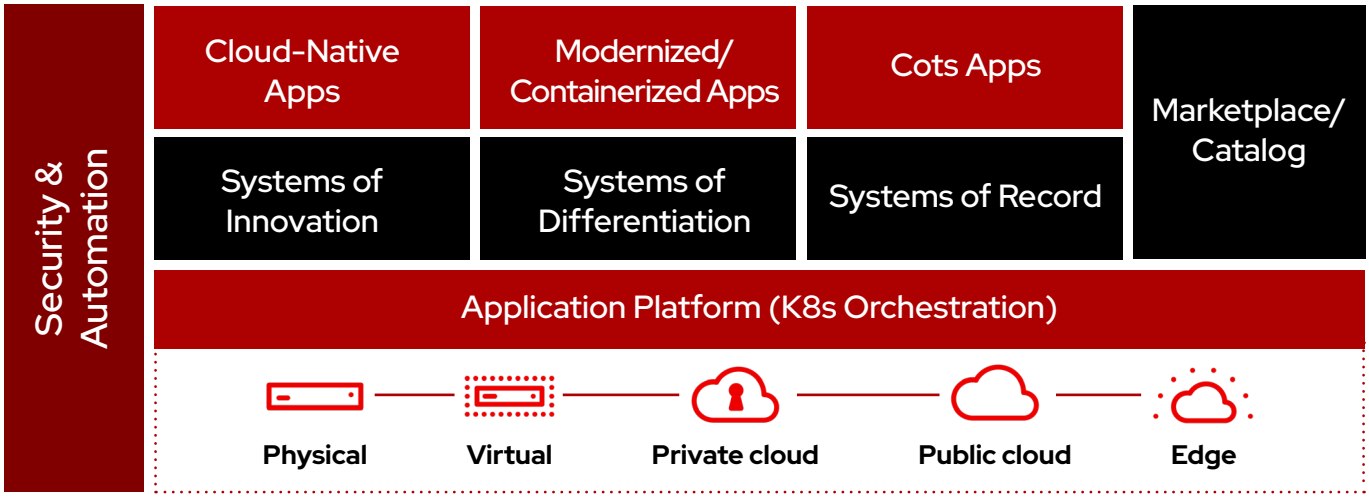


Figure 3: Support in Red Hat OpenShift for all phases of the application platform.

In addition to this highly scalable, flexible, and standards-based platform, Red Hat provides a platform, implemented as a software factory with a trusted software supply chain (TSSC), for moving code from source control to production as fast as possible, without compromising security and compliance (Figure 4).

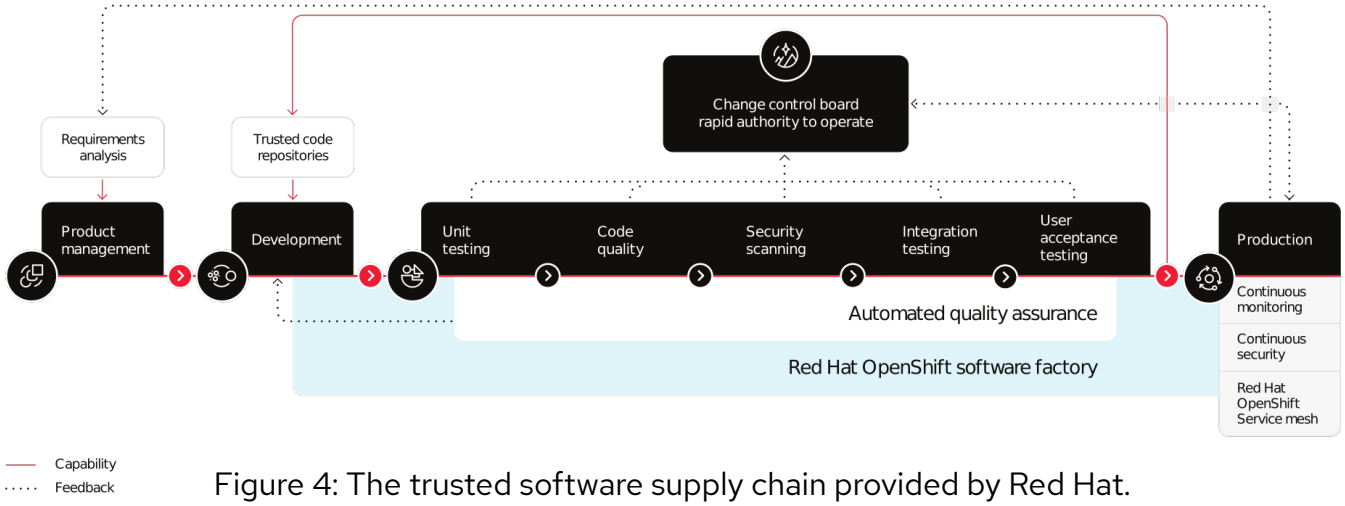


Figure 4: The trusted software supply chain provided by Red Hat.

Powered by Red Hat OpenShift, the TSSC brings together trusted third-party tools and prescriptive workflows for best practices such as test-driven development and CI/CD. The TSSC enforces best practices, for example, not allowing code into production before it has been validated with static code analysis and security scanning tools. The TSSC also makes the right action easy, for example, requiring developers to pull components (containers, libraries, binaries) from a trusted code repository. By enforcing best practices with opinionated gates and other controls (Figure 5), the TSSC provides a high degree of confidence in code deployments in an automated and secured manner.

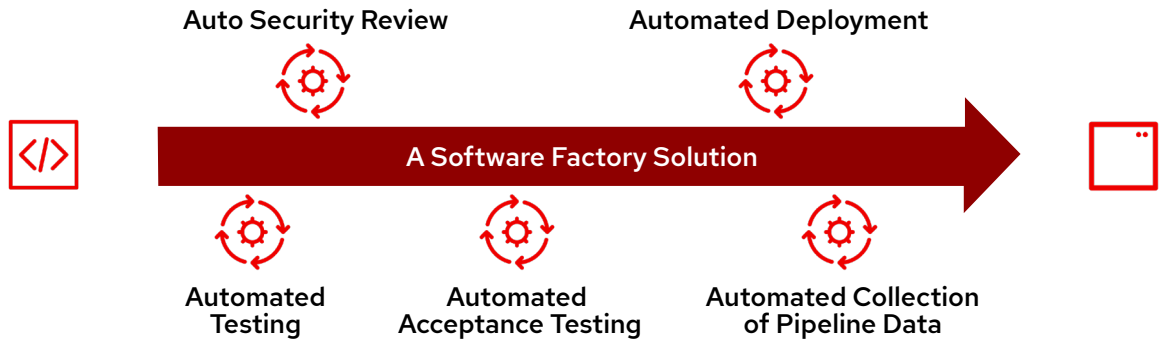
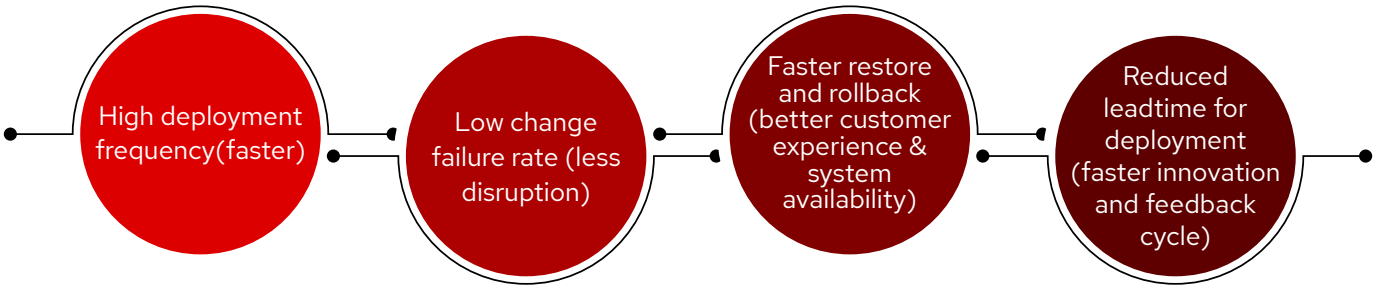


Figure 5: Steps in the software factory solution to trusted deployments.

The key benefits of a software factory built with TSSC are:



With Red Hat's open hybrid cloud architecture, founded on Red Hat's TSSC, organizations can build the foundation for a stable and efficient IDP that improves the developer experience and delivers better and faster business outcomes.

Red Hat offers an enterprise-grade, scalable, and secured platform along with a software factory to take the code to production in a consistent and automated way. Now let's double-click on some of the tools and capabilities that Red Hat provides to improve developer experience and make them more productive. The areas we would like to focus on are:

01 Application frameworks

02 Integrated Development Environment (IDE)

03 Backend tools and libraries

Application frameworks

As much as we like to focus on cloud-native application development, most enterprises are still traveling on their journey from traditional applications to cloud-native applications. To support this transition and to maximize the benefit of the existing applications and data ecosystem that exist in the enterprise, it is important for the developer platform to provide a rich choice of libraries, application frameworks, and tools that support both modern and traditional application development practices, allowing developers to connect, extend, and secure these applications.

Red Hat addresses this challenge with Red Hat Application Foundations. This offering is a rich set of development frameworks and tools (Figure 6) that help developers create modern as well as traditional applications on top of OpenShift. Application Foundations also includes:

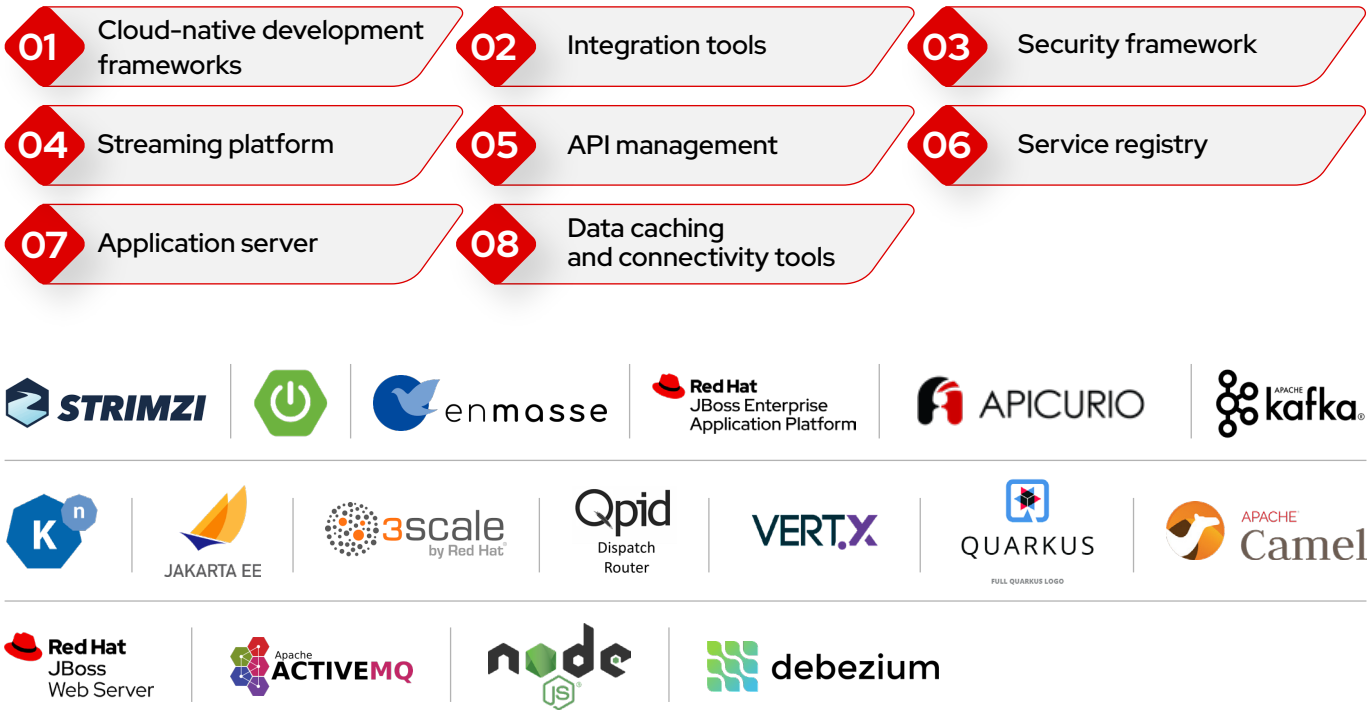


Figure 6: Some of the open source offerings in Red Hat Application Foundations.

With this rich set of capabilities, Red Hat Application Foundation delivers a single ecosystem for developers to build and manage a comprehensive enterprise application landscape consisting of modern cloud-native and traditional three-tier applications along with the required tools for integration, security, and data connectivity.

Red Hat also provides a migration toolkit that helps organizations convert their traditional Java EE/Jakarta workloads into containers to start their modernization journey. We see many organizations using the migration toolkit to re-platform their applications onto a hybrid cloud to standardize and improve operational efficiency at scale. Once the re-platformed application is

deployed on the hybrid cloud platform, developers can refactor and rebuild their code to make modern, cloud-native applications using cloud-native frameworks and tools such as Quarkus and Springboot provided in Application Foundations.

Integrated Development Environment (IDE)

In order to address the IDE challenges that we explained in the previous section, Red Hat provides a development environment/editor called Red Hat OpenShift Dev Spaces (formerly known as Red Hat CodeReady Workspaces). Red Hat believes in meeting developers where they are, to minimize disruption and maximize developer productivity. To this end, OpenShift Dev Spaces provides a faster and more reliable foundation on which to work with a shared development environment, offering a browser-based IDE as well as extension libraries and plugins that work with popular IDEs, for enterprises who would like to keep their existing IDE.

OpenShift Dev Spaces also gives operations teams more refined control to enforce organizational security and compliance standards. Some of the key benefits of OpenShift Dev Spaces are:

01 Near Instant on-boarding: Only a supported browser and access to a workspace is required to on-board a new developer.

02 Consistent and standardized configuration: Workspaces are centrally configured to ensure a consistent working environment across developers. A workspace includes project repositories, application runtimes, libraries, development tools, build tools, and resource allocation.

03 Browser-based IDE: OpenShift Dev Spaces includes a powerful and familiar in-browser IDE with support for Microsoft Visual Studio Code extensions. Developers need only a machine capable of running a web browser to code, build, test, and run on OpenShift.

For organizations who would like to keep their existing developer IDE ecosystem, Red Hat provides a comprehensive extensions/plugins to work with Application foundation for popular IDEs such as:

Visual Studio Code

JetBrains IDE (e.g., IntelliJ)

Eclipse IDE

Azure DevOps

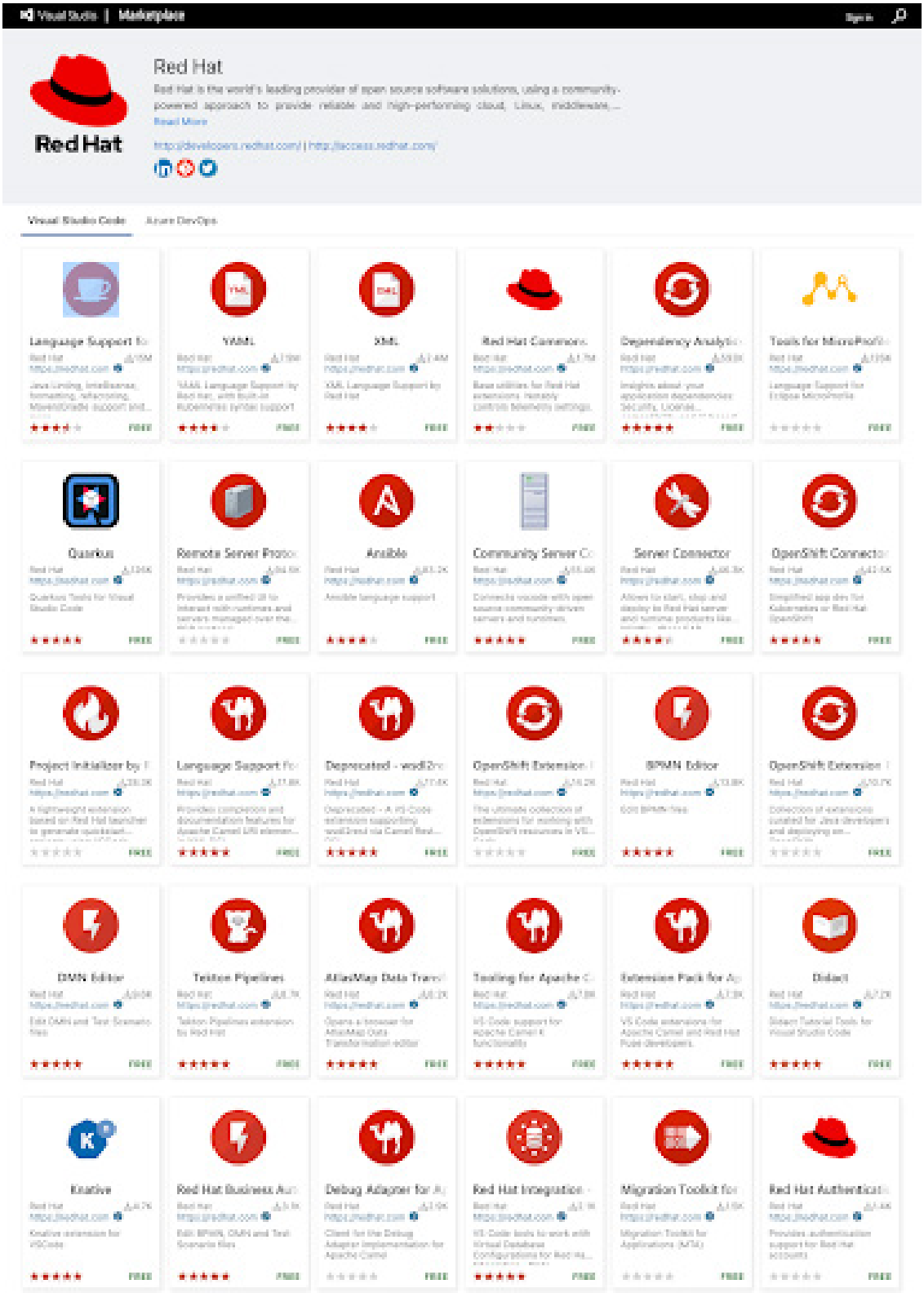


Figure 7: Visual Studio Marketplace

The OpenShift console includes a developer perspective that helps developers to

- Create and deploy applications on OpenShift.
- Visually interact with applications, components, and the services associated with them (Figure 7).

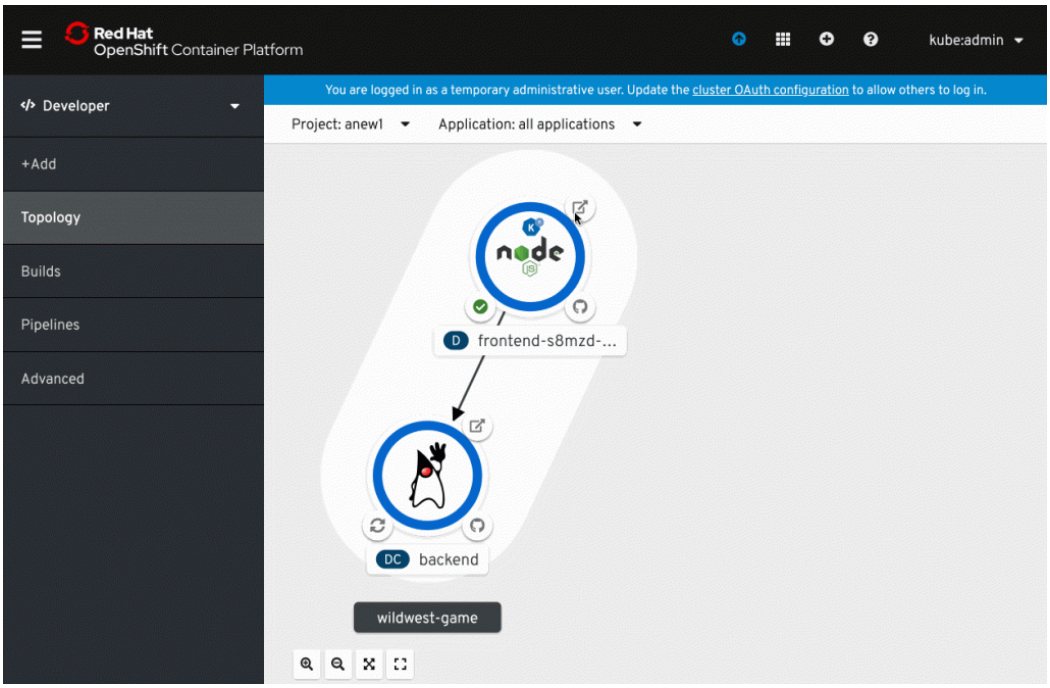


Figure 7: The topology view allows application management through a GUI.

- Find backend services, builder images, and templates for developer consumption in a developer catalog (Figure 8).

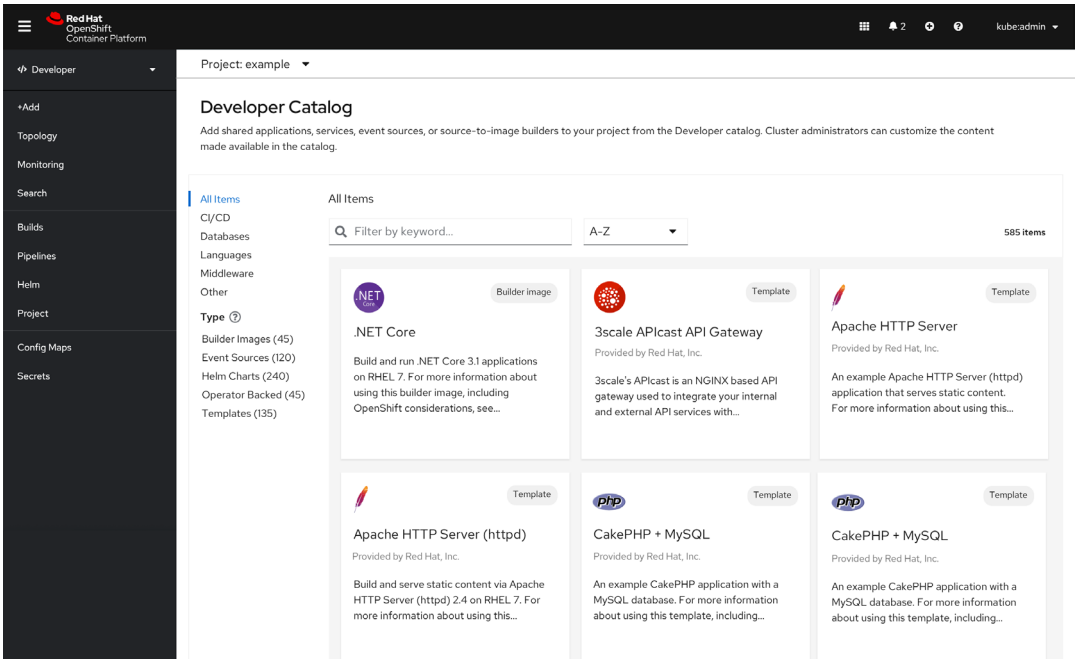


Figure 8: Developer catalog in OpenShift.

Monitor applications and take advantage of tools for observability (Figure 9).

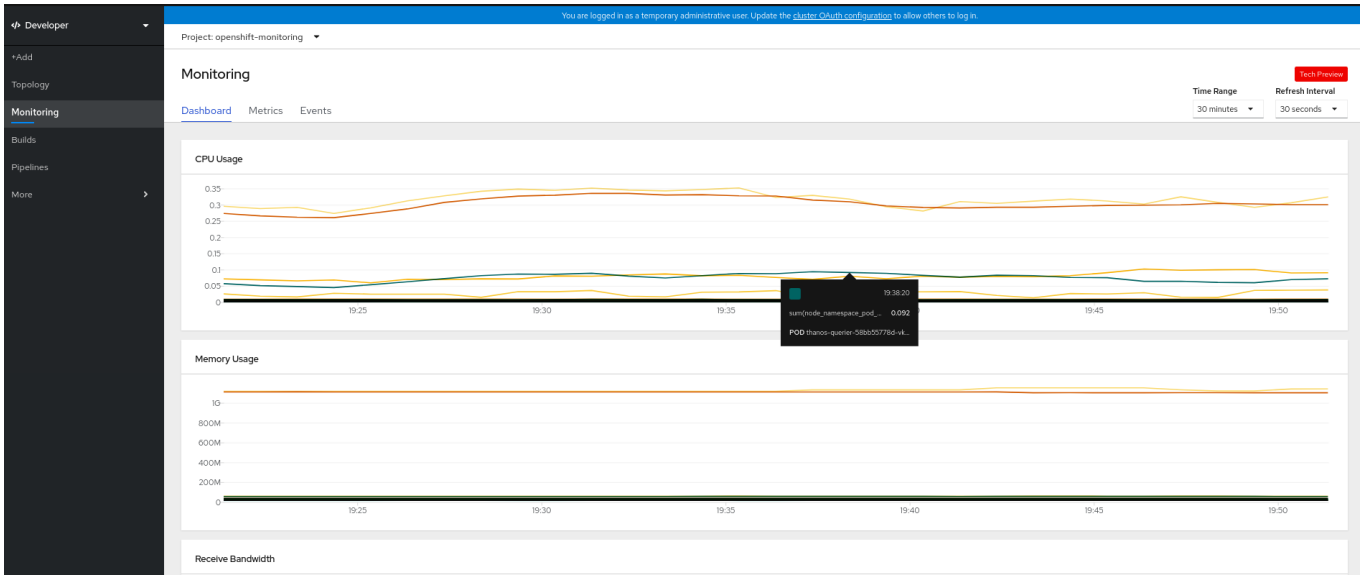


Figure 9: Monitoring portal in OpenShift.

Overall, Red Hat OpenShift Dev Spaces enables developers to focus more on delivering code at a faster and more consistent environment without sacrificing security and compliance standards. With the flexibility to retain an existing IDE or use OpenShift Dev Spaces with the developer perspective in the OpenShift console, Red Hat provides a comprehensive developer environment to improve developer experience and reduce cognitive load.

Backend tools and libraries

With its roots in open source innovation, Red Hat has been a key contributor to many of the popular free and open source tools with which enterprise developers and architects build and manage their cloud-native and traditional applications. From integration to API management to data access and caching, Red Hat supports open source tools and libraries for developers to address all their requirements for application development modernization.

Red Hat also hosts a comprehensive partner ecosystem from top global system integrators for consulting services, Technology providers to independent service vendors (ISVs) with certified software on Red Hat's platform and marketplace. Organizations can bring in their tools of choice from this ecosystem to customize and extend their platform to accommodate their business and developer requirements. Figure 10 shows some of Red Hat's top partners.

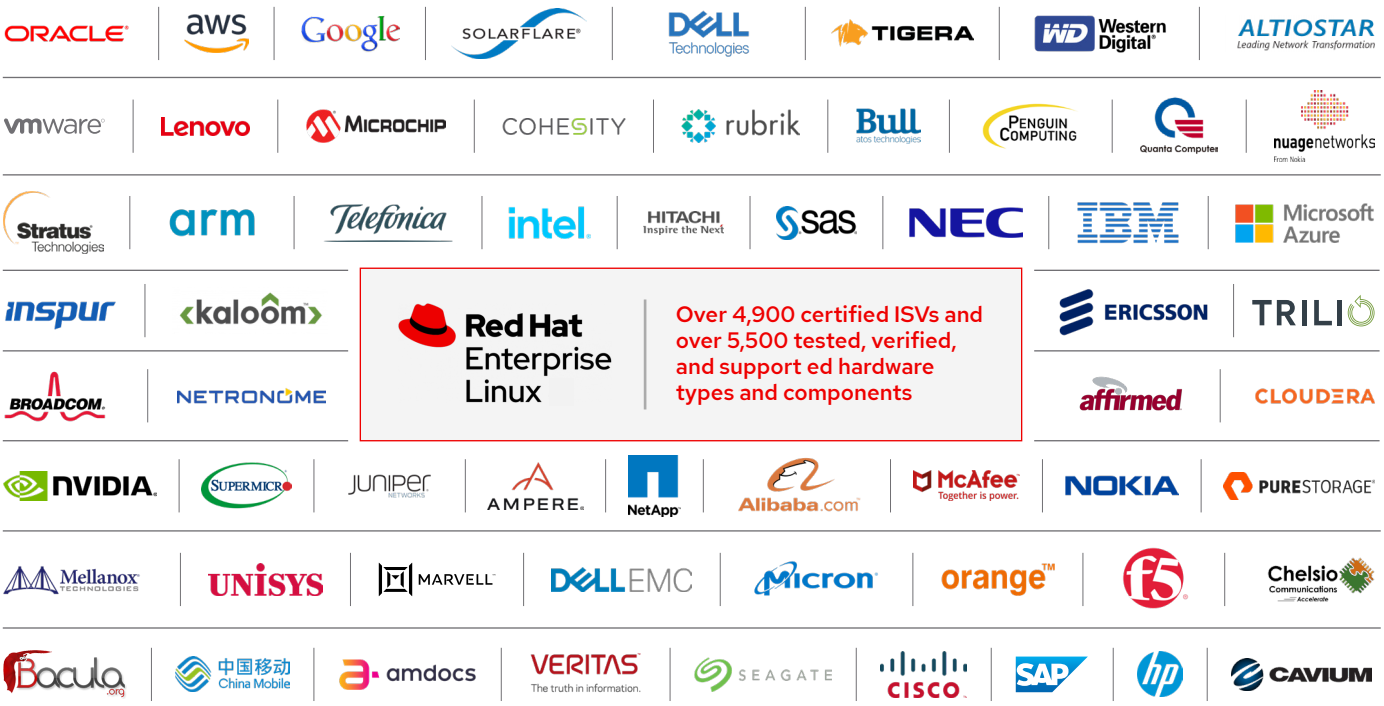


Figure 10: Some Red Hat partners.

In short, Red Hat provides a comprehensive developer ecosystem for enterprises on a stable and reliable open hybrid cloud platform.

Tying it all together

Even though the inner loop is where the developer experience and productivity focuses, let's not forget the importance of the platform and the outer loop. A stable and secured platform and a flexible and automated outer loop make sure developer innovations are taken to production as fast and as securely as possible to deliver desired business outcomes. With Red Hat OpenShift on an open hybrid cloud architecture across self-managed and managed cloud services offerings and a trusted software supply chain, Red Hat provides industry leading platform and outer loop capabilities for enterprises. For Inner loop, Red Hat provides a comprehensive ecosystem for developers to improve the overall productivity and reduce cognitive load through:

Figure 11 shows how Red Hat's cloud-based services all fit together.

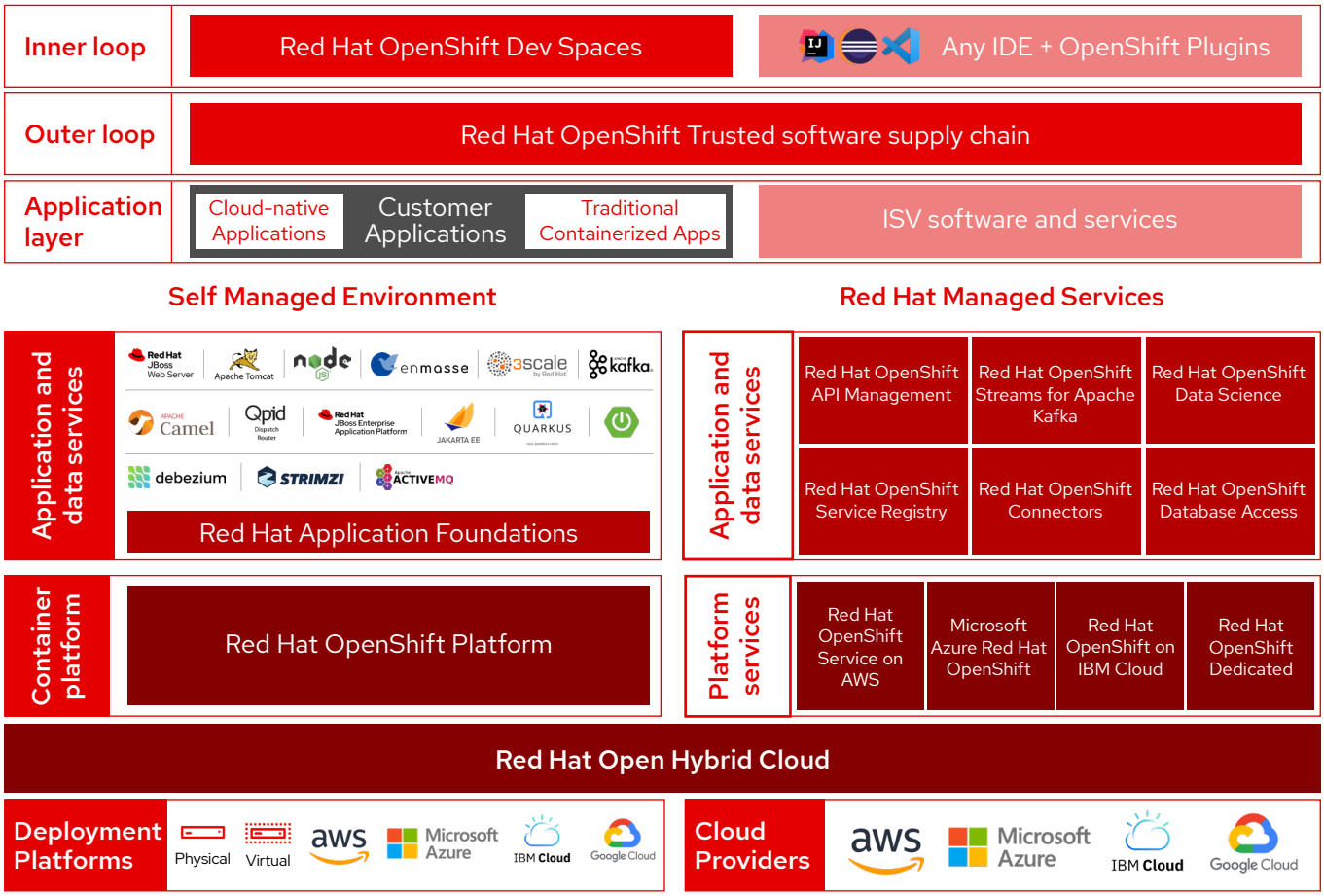


Figure 11: The ecosystem offered by Red Hat.

By adding an IDP as the frontend for their developers, enterprises can take advantage of the comprehensive ecosystem that Red Hat provides to deliver a highly innovative and collaborative developer ecosystem on a secure, scalable, and hybrid platform that will serve as a foundation for enterprises to thrive in the modern digital-led business environment.

References

- [1] IDC FutureScape: Worldwide Developer and DevOps 2022 Top 10 Predictions
- [2] https://martinfowler.com/articles/developer-effectiveness.html
- https://cloud.google.com/architecture/devops/devops-measurement-monitoring-and-observability#:~:text=Observability%20is%20tooling%20or%20a,patterns%20not%20defined%20in%20advance.
- https://www.redhat.com/rhdc/managed-files/ve-trusted-software-supply-chain-brief-f24751pr-202010-en.pdf
- https://martinfowler.com/articles/platform-prerequisites.html
- https://blog.container-solutions.com/internal-developer-platforms
- https://martinfowler.com/articles/developer-effectiveness.html